

### REMARKS

This responds to the Office Action mailed on March 30, 2007, and the references cited therewith.

Claims 1, 3-5, 7, 9, 11-16, 18, 20, and 22 were amended; as a result, claims 1-22 are now pending in this application.

#### Claim Objection

Claim 7 was modified to correct a typo as noted by the examiner.

#### §112 Rejection of the Claims

Claim 13 was modified to correct an incorrect dependency as noted by the examiner.

#### §101 Rejection of the Claims

Independent claims 1, 5, 7, 9, 11, 12, 16, 18, 20, and 22 have been amended to produce a useful, concrete, tangible result as required by 35 U.S.C. 101. Dependent claims 2-4, 6, 8, 10, 13-15, 17, 19, and 21 thus also now produce a useful, concrete, tangible result as required by 35 U.S.C. 101.

#### §103 Rejection of the Claims

Claims 1, 3, 12, and 14 were rejected under 35 U.S.C. § 103(a) as being unpatentable over “:-) When you grade that: using e-mail and the network in programming courses” by Arnow (hereinafter “Arnow”) in view of “Strings” by Worthington (hereinafter “Worthington”). The Examiner states:

Arnow does not expressly disclose: *e) putting each remaining line of functional programming code of the first file into an array of text strings; f) putting each remaining line of functional programming code of the second file into a second array of text strings*; However, Worthington teaches using arrays to store strings. See top of page 3, e.g. "It is often useful to store strings in arrays." It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Worthington's array

of strings to store Arnow's program code in order to utilize the useful nature of arrays as suggested by Worthington.

**Amended Claim 1 recites:**

"A method for comparing two program source code files to help an expert determine whether one file contains source code that has been copied from the other file or whether both files contain code that has been copied from a third file, the method comprising  
eliminating programming comments from the first source code file;  
eliminating programming comments from the second source code file;  
substituting a single space character for sequences of whitespace characters in each remaining line of functional programming code in said first file;  
substituting a single space character for sequences of whitespace characters in each remaining line of functional programming code in said second file;  
eliminating all lines of functional programming code in said first file that consist entirely of programming keywords;  
eliminating all lines of functional programming code in said second file that consist entirely of programming keywords;  
putting each remaining line of functional programming code of the first file into an first array of text strings;  
putting each remaining line of functional programming code of the second file into a second array of text strings; and  
finding all matches between text strings in said first array with text strings in said second array; and displaying a report showing said matches."

**Amended Claim 12 recites:**

"An apparatus for comparing two program source code files to help an expert determine whether one file contains source code that has been copied from the other file or whether both files contain code that has been copied from a third file, the apparatus comprising  
A computer;  
A source code matching program on said computer, wherein said source code matching program comprises:  
means for eliminating programming comments from the first source code file;  
means for eliminating programming comments from the second source code file;  
means for substituting a single space character for sequences of whitespace characters in each remaining line of functional programming code in said first file;  
means for substituting a single space character for sequences of whitespace characters in each remaining line of functional programming code in said second file;  
eliminating all lines of functional programming code in said first file that consist entirely of programming keywords;  
eliminating all lines of functional programming code in said second file that consist entirely of programming keywords;  
means for putting each remaining line of functional programming code of the first file into an first array of text strings;  
means for putting each remaining line of functional programming code of the second file into a second array of text strings; and  
means for finding all matches between text strings in said first array with text strings in said second array; and  
means for displaying a report showing said matches."

Claim 3 has been amended to be dependent on Claim 1. Amended Claim 3 recites:

~~“A method for comparing two program source code files to help an expert determine whether one file contains source code that has been copied from the other file or whether both files contain code that has been copied from a third file~~ The method of claim 1), the method further comprising  
eliminating functional programming lines from the said first source code file, leaving comment lines;  
eliminating functional programming lines from the said second source code file, leaving comment lines;  
substituting a single space character for sequences of whitespace characters in each remaining comment line in said first file;  
substituting a single space character for sequences of whitespace characters in each remaining comment line in said second file;  
putting each remaining comment line of the first file into an first array of text strings;  
putting each remaining comment line of the second file into a second array of text strings; ~~and~~  
finding all matches between text strings in said first array with text strings in said second array; ~~and~~  
displaying a report showing said matches.”

Claim 14 has been amended to be dependent on Claim 12. Amended Claim 14 recites:

~~“An apparatus for comparing two program source code files to help an expert determine whether one file contains source code that has been copied from the other file or whether both files contain code that has been copied from a third file~~ The apparatus of claim 12), the apparatus further comprising  
A computer;  
A source code matching program on said computer, wherein said source code matching program comprises:  
means for eliminating functional programming lines from the said first source code file, leaving comment lines;  
means for eliminating functional programming lines from the said second source code file, leaving comment lines;  
means for substituting a single space character for sequences of whitespace characters in each remaining comment line in said first file;  
means for substituting a single space character for sequences of whitespace characters in each remaining comment line in said second file;  
means for putting each remaining comment line of the first file into an first array of text strings;  
means for putting each remaining comment line of the second file into a second array of text strings;  
~~and~~  
means for finding all matches between text strings in said first array with text strings in said second array; ~~and~~  
means for displaying a report showing said matches.”

Neither Arnow nor Worthington teaches or suggests “eliminating all lines of functional programming code that consist entirely of programming keywords.” As such, Arnow in view of

Worthington fails to teach each and every element of claims 1, 3, 12, and 14. Therefore, Applicant respectfully submits that at least for this reason, independent claims 1 and 12 and dependent claims 3 and 14 are now allowable.

Claims 2, 4, 13, and 15 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Arnow in view of Worthington and further in view of "Plagiarism in natural and programming languages: an overview of current tools and technologies" by Clough (hereinafter "Clough"). The Examiner states:

In regard to claim 2, the above rejection of claim 1 is incorporated. Arnow and Worthington does not expressly disclose: *where finding all matches ignores the type case of the text*. However, Clough teaches that the YAP system translates upper-case letters to lower case letters. See page 24 under "Preprocess the submitted reports." It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Clough's case translation with Arnow's program code in order to preprocess text for tokenization as suggested by Clough.

Clough does not teach or suggest "eliminating all lines of functional programming code that consist entirely of programming keywords." As such, Arnow in view of Worthington and further in view of Clough fails to teach each and every element of amended claims 2, 4, 13, and 15. Therefore, Applicant respectfully submits that at least for this reason, dependent claims 2, 4, 13, and 15 are now allowable.

Claims 5, 6, 16, and 17 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Arnow in view of Clough and further in view of Worthington.

Amended Claim 5 recites:

"A method for comparing two program source code files to help an expert determine whether one file contains source code that has been copied from the other file or whether both files contain code that has been copied from a third file, the method comprising

extracting all words between whitespace from each line of functional programming code in the first source code file to an first array of text strings;

eliminating programming language keywords from said first array of text strings;

eliminating all words from said first array of text strings that are less than a minimum length of characters;

extracting all words between whitespace from each line of functional programming code in the second source code file to a second array of text strings;

eliminating programming language keywords from said second array of text strings;

eliminating all words from said second array of text strings that are less than a minimum length of characters;  
finding all matches between text strings in said first array with text strings in said second array; and  
displaying a report showing said matches.”

Amended Claim 16 recites:

“An apparatus for comparing two program source code files to help an expert determine whether one file contains source code that has been copied from the other file or whether both files contain code that has been copied from a third file, the apparatus comprising

A computer;

A source code matching program on said computer, wherein said source code matching program comprises:

means for extracting all words between whitespace from each line of functional programming code in the first source code file to an first array of text strings;

means for eliminating programming language keywords from said first array of text strings;

means for eliminating all words from said first array of text strings that are less than a minimum length of characters;

means for extracting all words between whitespace from each line of functional programming code in the second source code file to a second array of text strings;

means for eliminating all words from said second array of text strings that are less than a minimum length of characters;

means for eliminating programming language keywords from said second array of text strings;

means for finding all matches between text strings in said first array with text strings in said second array; and

means for displaying a report showing said matches.”

Neither Arnow nor Clough nor Worthington teaches or suggests “eliminating all words from said second array of text strings that are less than a minimum length of characters.” As such, Arnow in view of Clough and further in view of Worthington fails to teach each and every element of claims 5 and 16. Therefore, Applicant respectfully submits that at least for this reason, independent claims 5 and 16 and dependent claims 6 and 17 are now allowable.

Claims 7, 8, 18, and 19 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Arnow in view of Clough and further in view of Worthington. The Examiner states:

In regard to claim 7, Arnow and Worthington do not expressly disclose: *e) finding all partial matches between text strings in said first array with text strings in said second array, where a partial match is where one string can be found in its entirety in as a second string but the strings are not identical.* However, Clough teaches matching substrings. See page 22, bulleted list in section 4.1.3, e.g. “q as a substring of S.” It would have been obvious to one of ordinary skill at the time the invention was made, to use

Clough's teaching of substring matching with Arnov's plagiarism detector in order to detect systemic changes to variable names as suggested by Clough (see 1" paragraph in section 4.1.3).

However, Clough states on page 22 in the preceding paragraph from the one that the Examiner quotes, the following:

DUP is able to identify p-matches by replacing actual tokens such as identifiers by their offsets. This removes identif[iers], but preserves the distinctness of different identifiers. The first occurrence of an identifier is replaced by 0, later ones by the number of tokens since the previous one. For example, given two code segments:  $u(x,y)=(x>y)?x:y$  and  $u(w,z)=(w>z)?w:z$ , both would be encoded as  $0(0,0)=(6>6)?5:5$  because u and v occur in the same positions, as do pairs x and w and y and z.

Thus Clough teaches finding all partial matches of lines of code that have been modified from their original form but does not teach extracting identifiers and finding all partial matches of identifiers that have not been modified from their original form. Not only does Clough not disclose or suggest finding all partial matches of identifiers, Clough teaches removing all identifiers from the code and replacing them with numbers for this comparison, thus teaching away from Claims 7, 8, 18, and 19. Applicant therefore concludes that, because Clough teaches away from Claims 7, 8, 18, and 19, Claims 7, 8, 18, and 19 are each allowable over Arnov in view of Clough and further in view of Worthington.

Claim 11 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Arnov in view of Clough. Applicant assumes that the Examiner meant to reject Claim 22 for the same reason though no rejection of Claim 22 under 35 U.S.C. § 103(a) was given. The Examiner states:

Arnov does not expressly disclose *wherein the first and second sets of code elements are selected from the group consisting of complete words, selected partial words*. However, Clough teaches searching for complete and partial words. See bottom of page 11 "show the number of words in common." It would have been obvious to one of ordinary skill at the time the invention was made, to use Clough's teaching of word matching with Arnov's keyword elimination in order to identify documents that

contain similar passages; as suggested by Clough (see bottom of page 11). Also see page 22, bulleted list in section 4.1.3, e.g. "q as a substring of S." It would have been obvious to one of ordinary skill at the time the invention was made, to use Clough's teaching of substring matching with Arnow's plagiarism detector in order to detect systemic changes to variable names as suggested by Clough (see 1<sup>st</sup> paragraph in section 4.1.3).

As stated previously, Clough teaches finding all partial matches of lines of code that have been modified from their original form but does not teach extracting identifiers and finding all partial matches of identifiers that have not been modified from their original form. Not only does Clough not disclose or suggest finding all partial matches of identifiers, Clough teaches removing all identifiers from the code and replacing them with numbers for this comparison, thus teaching away from Claims 11 and 22. Applicant therefore concludes that, because Clough teaches away from Claims 11 and 22, Claims 11 and 22 are each allowable over Arnow in view of Clough.

Claims 9, 10, 20 and 21 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Arnow in view Worthington and in further view of "TLA+ Mode: Editing Features" by Wegmann (hereinafter "Wegmann"). The Examiner states:

In regard to claim 9, Arnow discloses using keywords to find matching code. See page 13, 2nd column, 2nd paragraph, e.g. "the other only keywords and operators." Arnow and Worthington do not expressly disclose: *g) finding sequences where the first word of each line in said first array matches the first word of each line in said second array*. However, Wegmann teaches that keywords are found at the beginning of a line. See page 4, 3<sup>rd</sup> paragraph under "Formatting and Indenting," e.g. "The keywords whose templates can be inserted all start at the very beginning of a line." It would have been obvious to one of ordinary skill at the time the invention was made, to use Wegmann's teaching of the location of keywords with Arnow's keyword matching in order to easily find the likely location of keywords as suggested by Wegmann.

Applicant disagrees with the Examiner for several reasons. First, no mention is made in claims 9, 10, 20 and 21 about keywords and so the teachings of Wegmann do not apply to these

claims. Second, Wegmann states on page 4, 3<sup>rd</sup> paragraph under "Formatting and Indenting," that "The keywords whose templates can be inserted all start at the very beginning of a line..." This reference refers to a particular kind of keyword but does not claim that all keywords are located at the beginning of a line. Third, TLA+ is a particular programming language with a very specific syntax as described in "On the Logic of TLA+" by Merz (hereinafter "Mertz"). Such a particular programming language should not be generalized to all programming languages and would not be so generalized by one of ordinary skill in the art. Fourth, Wegmann refers to a special mode of operation of the GNU Emacs text editor as described in Wegmann on the cover page in the first paragraph under the section entitled "Copying" and again on page 6 in the first paragraph under the section entitled "Outlining." Emacs is further described in the Wikipedia entry for "Emacs" (hereinafter "Wikipedia"). Emacs is a text editor and has no particular use in finding plagiarism or comparing sequences of lines of code. For each of these reasons, Applicant concludes that Wegmann does not teach any part of Claims 9, 10, 20 and 21 and are each allowable.



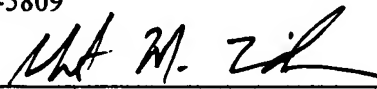
**CONCLUSION**

Applicant respectfully submits that the claims are in condition for allowance, and notification to that effect is earnestly requested. The Examiner is invited to telephone Applicant at 408-741-5809 to facilitate prosecution of this application.

Respectfully submitted,

ROBERT M. ZEIDMAN  
15565 Swiss Creek Lane  
Cupertino, CA 95014  
408-741-5809

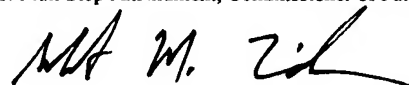
Date APRIL 24, 2007

By   
Robert M. Zeidman  
Customer No. 65069

**CERTIFICATE UNDER 37 CFR 1.8:** The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Mail Stop Amendment, Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this 13 day of MAY 2007.

ROBERT M. ZEIDMAN

Name



Signature